REMARKS

This is in response to the Final Action mailed December 29, 2003. The Office Action rejected claims 30, 39, and 52-53 under 35 U.S.C. § 102, and claims 40-44 under 35 U.S.C. § 103.

To more clearly claim that which Applicants consider a novel aspect of the invention, independent claims 30, 39, 52, and 53 have been amended to recite "generating a virtual machine instruction sequence by compiler to be executed by a virtual machine." This amendment incorporates language previously found in the preamble of independent claims 30, 39, 52, and 53 and also satisfies the Examiner's objections noted in the Advisory Action of March 29, 2004. As noted in more detail in the remarks below, this limitation is not found in the cited prior art. This amendment is made based upon the interview held with the Examiner on April 7, 2004, who indicated that such limitation had not been previously considered because it was found in the preamble of the claims.

Reconsideration in light of the amendments and remarks made herein is respectfully requested.

Rejections Under 35 U.S.C. § 102

The Office Action rejected claims 30, 39, 52, and 53 under 35 U.S.C. § 102(e) as being anticipated by Adl-Tabatabai (hereinafter Adl-Tabatabai) (U.S. Patent No. 6,170,083).

Applicants submit that the present claimed invention is patentably distinguishable from the teachings in Adl-Tabatabai. In particular, the prior art requires generation of basic blocks during both compile time and runtime. In the present claimed invention basic blocks are

generated only during compile time and transferred to the virtual machine to be used during runtime.

Adl-Tabatabai relates to Java™, as described in Col. 1, lines 4-7 of its specification. Generally speaking, when translating a program code into a bytecode, a Java compiler generates basic blocks and translates the program code into bytecode based on the generated basic blocks. The basic blocks generated by the compiler are destroyed upon generation of the bytecode for a class file. Because of this, in Adl-Tabatabai, basic blocks must be regenerated twice, once in the compile time and once in the runtime.

According the "Glossary of Java Related Terms" by Sun Microsystems, Inc., (http://java.sun.com/docs/glossary.html) "bytecode" and related terms are defined as follows.

> **bytecode:** Machine-independent code generated by the Java compiler and executed by the Java interpreter.
>
> **compiler:** A program to translate source code into code to be executed by a computer. The Java compiler translates source code written in the Java programming language into bytecode for the Java virtual machine.
>
> **interpreter:** A module that alternatively decodes and executes every statement in some body of code. The Java interpreter decodes and executes bytecode for the Java™ virtual machine.
>
> **Java™ virtual machine:** A software 'execution engine' that safely and compatibly executes the byte codes in Java class files on a microprocessor (whether in a computer or in another electronic device).

Furthermore, in Chapter I of "The Java™ Language Specification, Second Edition" (ISBN: 0-201-31008-2) by James Gosling and others of Sun Microsystems Inc., there is a

description that "bytecode" in a Java program is generated in "compile time" and executed in "runtime activities" (page 1, lines 11-15).

Compile time normally consists of translating programs into a machine-independent bytecode representation. In contrast, Runtime activities include: loading and linking of the classes needed to execute a program, optional machine code generation and dynamic optimization of the program, and actual program execution.

Applicant submits that Adl-Tabatabai relates to procedures executed in the "runtime activities", as defined by "The Java™ Language Specification, Second Edition", page 1, line 12 in Chapter 1, and the description in Chapter 12 "Execution" of the same book.

Moreover, according to the "Java™ Machine Specification" (ISBN: 0-201-63452-X) by Tim Lindholm and Frank Yellin published in 1996, information about file name and line number, among information in source code to be compiled, can be transmitted to the virtual machine in a class file using "Source File attribute" ("Java™ Machine Specification", Chapter 4, Section 7.2) and "Line Number Table attribute" ("Java™ Machine Specification", Chapter 4, Section 7.6). However, it is also described that "Source File attribute" and "Line Number Table attribute" are not essential. It describes that both "Source File attribute" and "Line Number Table attribute" are not able to store information for the basic blocks and the source code.

Further, in "Java Virtual Machine Instruction Set", Chapter 6, the description about bytecode states that there is no bytecode instruction for specifying the basic blocks. Also, in a description for "compile time", "Java Virtual Machine Instruction Set", Chapter 6, states that there is no bytecode instruction for specifying the basic blocks.

Applicants further submit the following details about conventional translating programs for translating a program into a machine-independent byte code representation in the compile time.

According to "Compilers: Principles, Techniques, and Tools" (ISBN: 0-201-10088-6) by A.V. Aho and others, a compiler is constituted by an order illustrated in Fig. 1.9 in Section 3 "The Phase of a Compiler" in Chapter 1 "Introduction to Compiling". A "code generator" in this section is detailed in Chapter 9 "Code Generation", a method for consisting the basic blocks is detailed in the Section 4, "Basic Block and Flow Graphs" in Chapter 9, and details about the generation of a target code that is outputted by the compiler are in Section 6 "A Simple Code Generator" in Chapter 9.

Applicants note the following description in "A Code-Generation Algorithm", Chapter 9, Section 6, - "The code-generation algorithm tasks as input a sequence of three-address statements consisting a basic block."

From this description, a Java language compiler, such as a javac used in the "compile time", generates basic blocks internally in order to translate a "program" and perform a conversion process into "bytecode" based on the generated basic blocks. The above sequence in the compiler is well known to those skilled in the art.

By checking the entire procedures from compiling of a program language in the "compile time" to the execution of the program in the "runtime activities", the following becomes clear.

In the conventional art, as typified by Adl-Tabatabai, an operation of division into basic blocks is performed twice, first in the "compile time", and second in the "runtime activities". In the "compile time", the division into the basic blocks is performed in order to translate the program. An operation in the "runtime activities" is the operation described in Col. 4 and Step 430 in Fig. 4 of the specification of Adl-Tabatabai.

The second division into the basic blocks in the "runtime activities" in Adl-Tabatabai becomes necessary because, as clear from the aforementioned specification of bytecode and class file of Java, information of the basic blocks generated in the translation in the "compile time" is lost when converted into bytecode for the class file.

On the other hand, independent claims 30, 39, 52, and 53 of the present invention literally or substantively claim "generating a virtual machine instruction sequence by compiler to be executed by a virtual machine; dividing the virtual machine instruction sequence into basic blocks each corresponding to an instruction block; transmitting the instruction block [basic blocks] to the virtual machine...." As described in the fifth embodiment of the present specification, a virtual machine compiler 7660 in Fig. 77 performs translation in the "compile time", and a decoding unit 3802, an execution unit 3810, and a memory stack 4422 perform an operation in the "runtime activities".

Fig. 79 of the present application specification shows a flowchart of the operation by a virtual machine compiler 7663 illustrated in Fig. 77. As shown in Step 7607 in the flowchart, the compiler of the present invention performs the division into the basic blocks at least once in the "compile time".

However, in the virtual machine illustrated by Fig. 71, the operation of division into the basic blocks is not performed in the "runtime activities", as shown by Figs. 72-76 and "construction of virtual machine" and "operation of virtual machine" in the fifth embodiment.

Further, the virtual machines illustrated by Figs. 85, 90, and 94, which are improved examples of the virtual machine of Fig. 71 or a JIT compiler illustrated in Fig. 99, do not include an operation or a step of division into basic blocks in the "runtime activities".

In summary, while Adl-Tabatabai needs to perform the division into the basic blocks twice, both in the "compile time" and in the "runtime activities", the present claimed invention does not have to perform the division into the basic blocks in the "runtime activities" at all, even if the division into the basic blocks is performed more than once in the "compile time". As recited in the independent claims of the present invention, the compiler generates the basic

blocks and transmits them to the virtual machine. No recreation or generation of the basic block is required by the virtual machine during execution.

This is possible because the present invention is designed so that a result of the division into the basic blocks in the "compile time" is transmitted to the virtual machine. Thus making it unnecessary to perform the division into the basic blocks again in the "runtime activities". However, in Adl-Tabatabai, information about the basic blocks is lost when the program code is converted into bytecode for the class file, and such basic blocks need to be recreated in the "runtime activities". In the present claimed invention, on the other hand, basic blocks generated in the compile time are not destroyed, but are instead transmitted to the virtual machine. Accordingly, there is no need to generate or recreate the basic blocks in the runtime again.

Naturally, in the present claimed invention the load during the "runtime activities" is reduced. Thus, it is apparent that a virtual machine according to the present claimed invention has a higher performance in comparison to Adl-Tabatabai's virtual machine.

Therefore, according to the present invention, it is possible to reduce the load of procedures during the runtime, and to realize a high-speed virtual machine in comparison with Adl-Tabatabai.

For at least these reasons, Applicants submit that claims 30, 39, and 52-53 are patentably distinguishable from the cited prior art.

Applicants respectfully request that the 35 U.S.C. § 102(e) rejection be withdrawn.

Rejections Under 35 U.S.C. § 103

The Office Action rejected claims 40, 42, and 44 under 35 U.S.C. § 103 as being unpatentable over Adl-Tabatabai (U.S. Patent No. 6,170,083) in view of Prosser et al. (hereinafter Prosser) (U.S. Patent No. 6,301,652).

The Office Action also rejected claims 41 and 43 under 35 U.S.C. § 103 as being unpatentable over Adl-Tabatabai in view of Prosser and further in view of Wahbe (U.S. Patent No. 6,151,618).

For the reasons cited above, Applicants submit that dependent claims 40, 42, and 44 are patentably distinguishable over the cited prior art. In particular, the prior art requires generation of basic blocks during both compile time and runtime. However, this prior art fails to teach or suggest that the basic blocks may be generated only during compile time and transferred to the virtual machine to be used during runtime as claimed.

Applicants respectfully request that the 35 U.S.C. § 103 rejection be withdrawn.

Conclusion

In view of the amendments and remarks made above, it is respectfully submitted that the pending claims are in condition for allowance, and such action is respectfully solicited. Authorization is hereby given to charge our Deposit Account No. 19-2814 for any charges that may be due. Furthermore, if an extension is required, then Applicants hereby request such an extension.

Respectfully submitted,

Snell & Wilmer L.L.P.

Julio Loza

Julio M. Loza
Registration. No. 47,758
1920 Main Street
Suite 1200
Irvine, CA 92614
Telephone: (949) 253-4924

I hereby certify that this document is being deposited on April 15, 2004 with the U.S. Postal Service as first class mail under 37 C.F.R. 1.8 and is addressed to, Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450.

By: James Lee

James

Signature

Dated: April 15, 2004